**Building a Comprehensive Honeypot on AWS: A Deep Dive**

Hands-on experience is crucial in cybersecurity, and creating a honeypot is one of the best ways to learn about attacker behavior and defense mechanisms. In this project, I deployed a honeypot on AWS to simulate real-world attacks and observe how attackers interact with the system. This detailed blog will walk you through each configuration and insight gained.

**What is a Honeypot?**

A honeypot is a decoy system designed to attract attackers, log their activities, and analyze their methods. Unlike real systems, honeypots are intentionally vulnerable, allowing researchers and cybersecurity professionals to learn about attack patterns and improve defenses without compromising actual production environments.

**Why Use AWS?**

AWS (Amazon Web Services) provides an ideal platform for deploying honeypots due to its scalability, accessibility, and robust security features. Here's why AWS stands out:

1. **Scalability**: Easily adjust resources to suit project needs.
2. **Accessibility**: Public IPs allow real-time attack simulations.
3. **Security**: Flexible security groups enable precise traffic control.
4. **Cost-Effectiveness**: The free-tier allows beginners to experiment without incurring costs.

**Project Goals**

The main objectives of this project were:

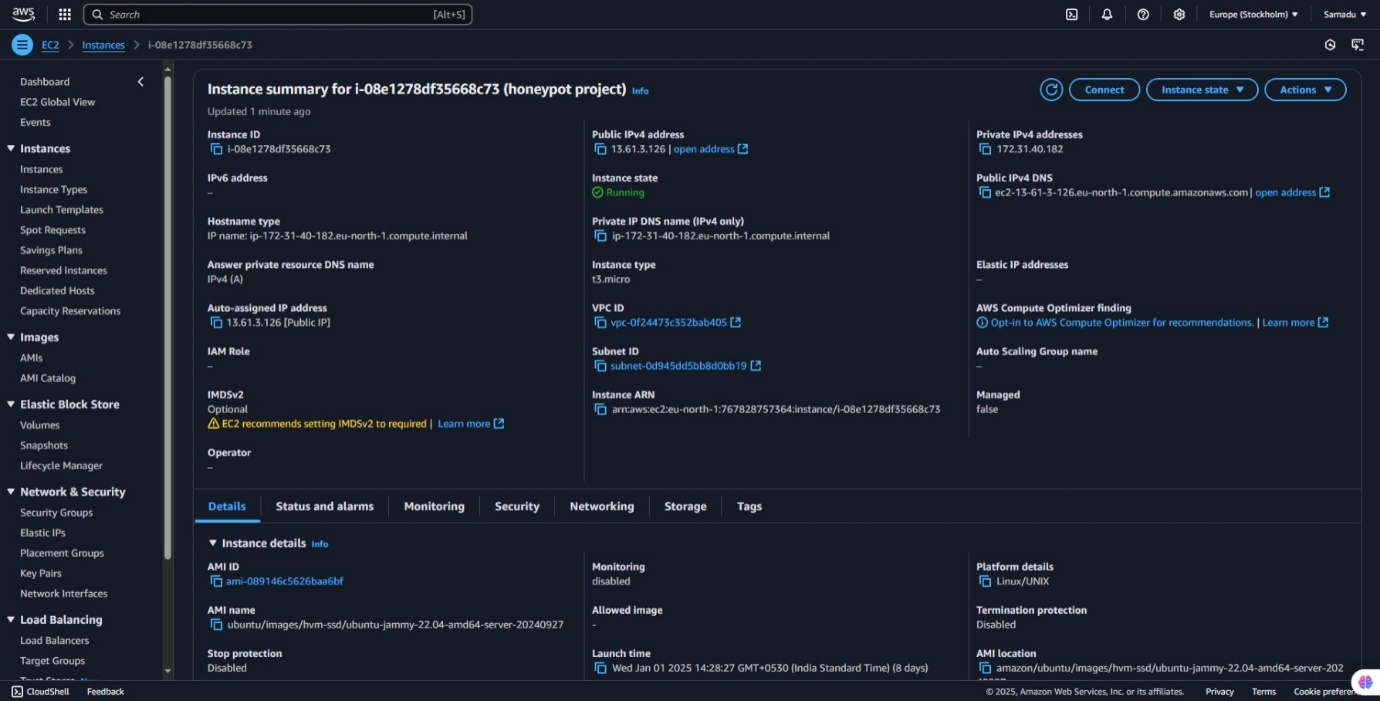
1. Deploy a honeypot on AWS to attract attackers.
2. Capture and analyze logs to understand attack methods.
3. Simulate an attack-defense scenario with designated roles:
   * **Person 1**: Configured and managed the honeypot.
   * **Person 2**: Played the role of the attacker.
   * **Person 3**: Acted as the defender, monitoring and mitigating threats remotely.

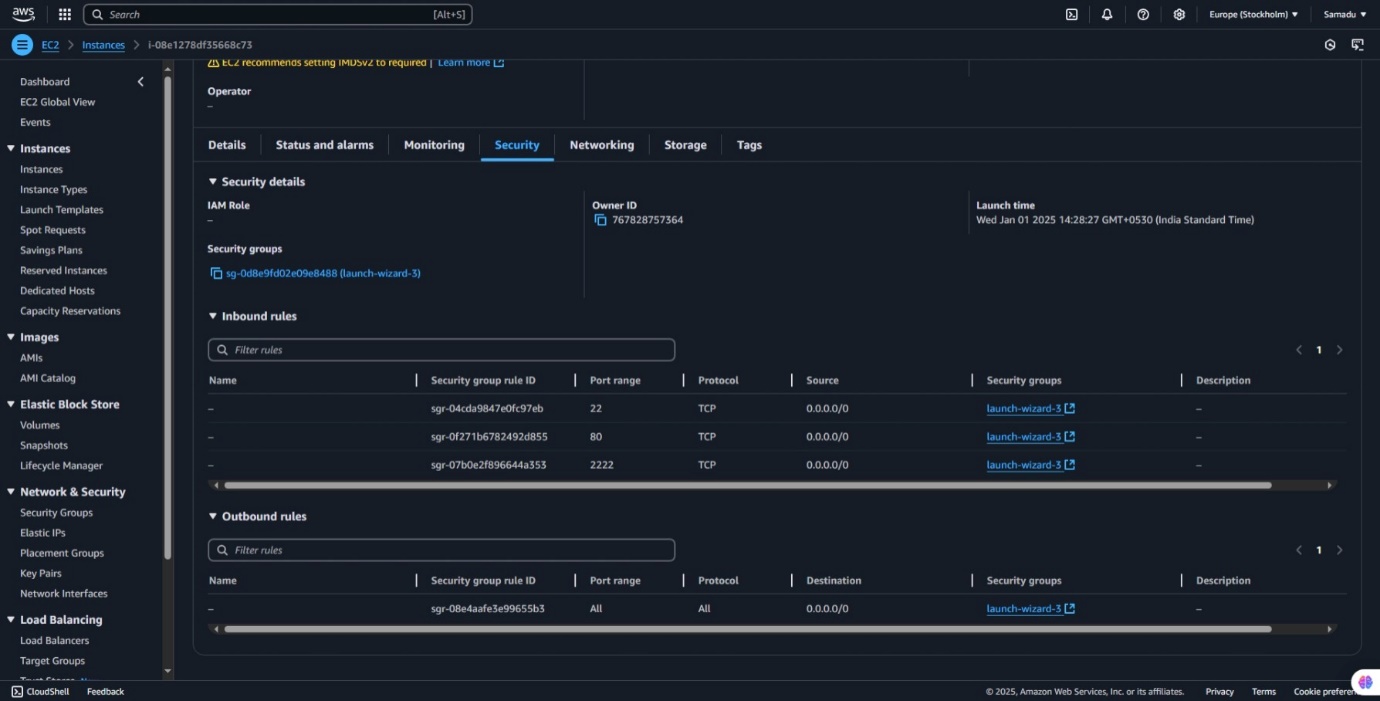
**Step-by-Step Deployment**

**1. Launching an AWS EC2 Instance**

AWS EC2 (Elastic Compute Cloud) allows you to create virtual machines. Here's how I set up the instance:

1. **Log in**: Access the [AWS Management Console](https://aws.amazon.com/console/).
2. **Create Instance**:
   * Selected **Ubuntu Server 22.04 LTS** as the operating system.
   * Chose **t2.micro** as the instance type (free-tier eligible).
3. **Security Group Configuration**:
   * Allowed inbound traffic on:
     + **Port 2222** for the honeypot.
     + **Port 22** for administrative SSH (restricted to my IP).
4. **Key Pair**: Generated a key pair for secure access and downloaded the .pem file.
5. **Launch**: Started the instance and noted its **public IPv4 address**.





**2. Setting Up Cowrie Honeypot**

Cowrie is a popular SSH/Telnet honeypot that logs all interactions. Here’s the setup process:

**Step 1: Connect to the Instance**

SSH into the instance using the key pair:

ssh -i <key-pair-name>.pem ubuntu@<public-ip>

**Step 2: Update System and Install Dependencies**

Ensure the system is up-to-date:

sudo apt update && sudo apt upgrade -y

sudo apt install git python3-venv python3-pip -y

**Step 3: Clone and Configure Cowrie**

1. Clone the repository:
2. git clone https://github.com/cowrie/cowrie.git
3. cd cowrie
4. Set up a virtual environment:
5. python3 -m venv cowrie-env
6. source cowrie-env/bin/activate
7. Install dependencies:
8. pip install -U pip setuptools
9. pip install -r requirements.txt
10. Configure Cowrie:
11. cp etc/cowrie.cfg.dist etc/cowrie.cfg
12. nano etc/cowrie.cfg
    * Set the listening port to 2222:
    * [ssh]
    * listen\_port = 2222

**Step 4: Start Cowrie**

Start the honeypot:

./bin/cowrie start

Verify its status:

./bin/cowrie status

**Step 5: Test the Setup**

Attempt to connect to the honeypot:

ssh -p 2222 anyuser@<public-ip>

This interaction will be logged by Cowrie.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Simulating an Attack**

**Role of the Attacker**

**Tools Used**:

* **Hydra**: For brute-force password attempts.
* **Metasploit Framework**: To exploit vulnerabilities.

**Steps**:

1. Identified the honeypot’s IP and port (2222)
2. Used Hydra for brute-forcing:
3. hydra -l root -P passwords.txt ssh://<public-ip>:2222
4. Exploited vulnerabilities with Metasploit:
5. msfconsole
6. use exploit/unix/ssh/sshexec
7. set RHOST <public-ip>
8. set RPORT 2222
9. run
10. Observed and documented the honeypot’s responses.

**Role of the Defender**

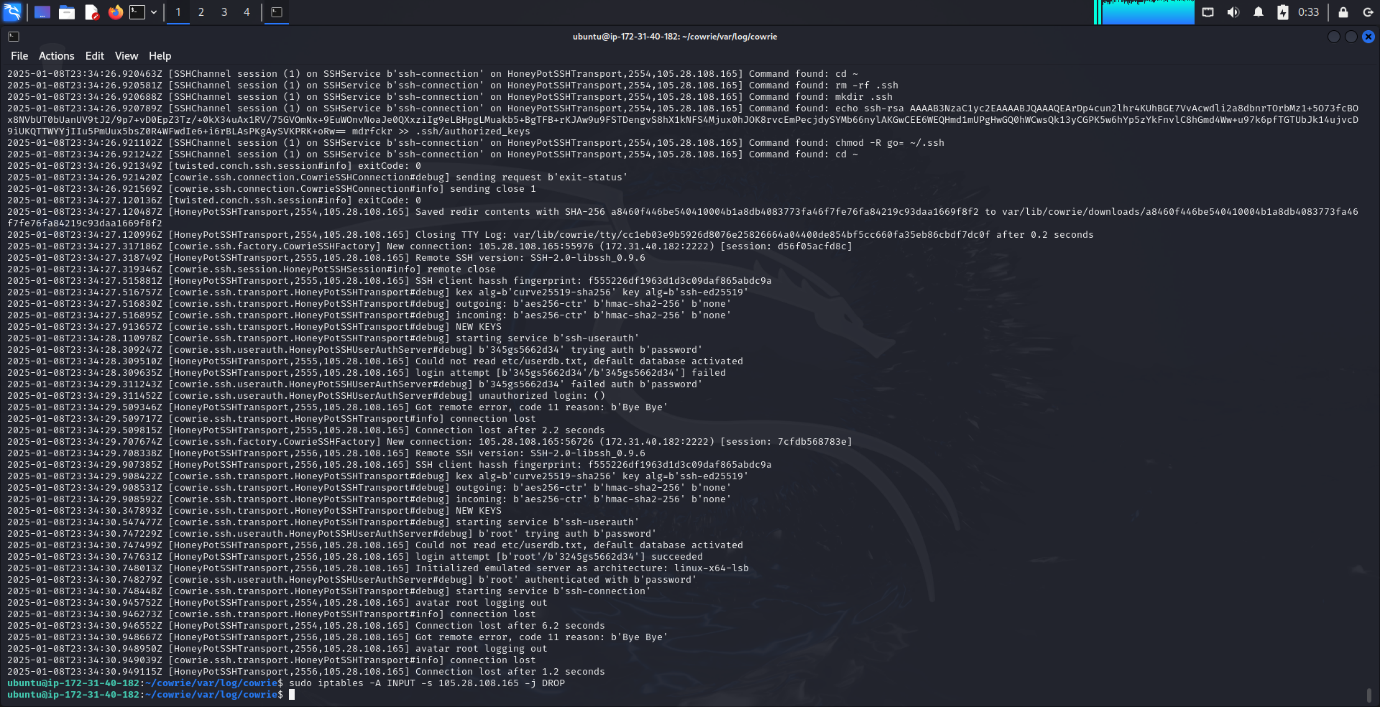
**Tools Used**:

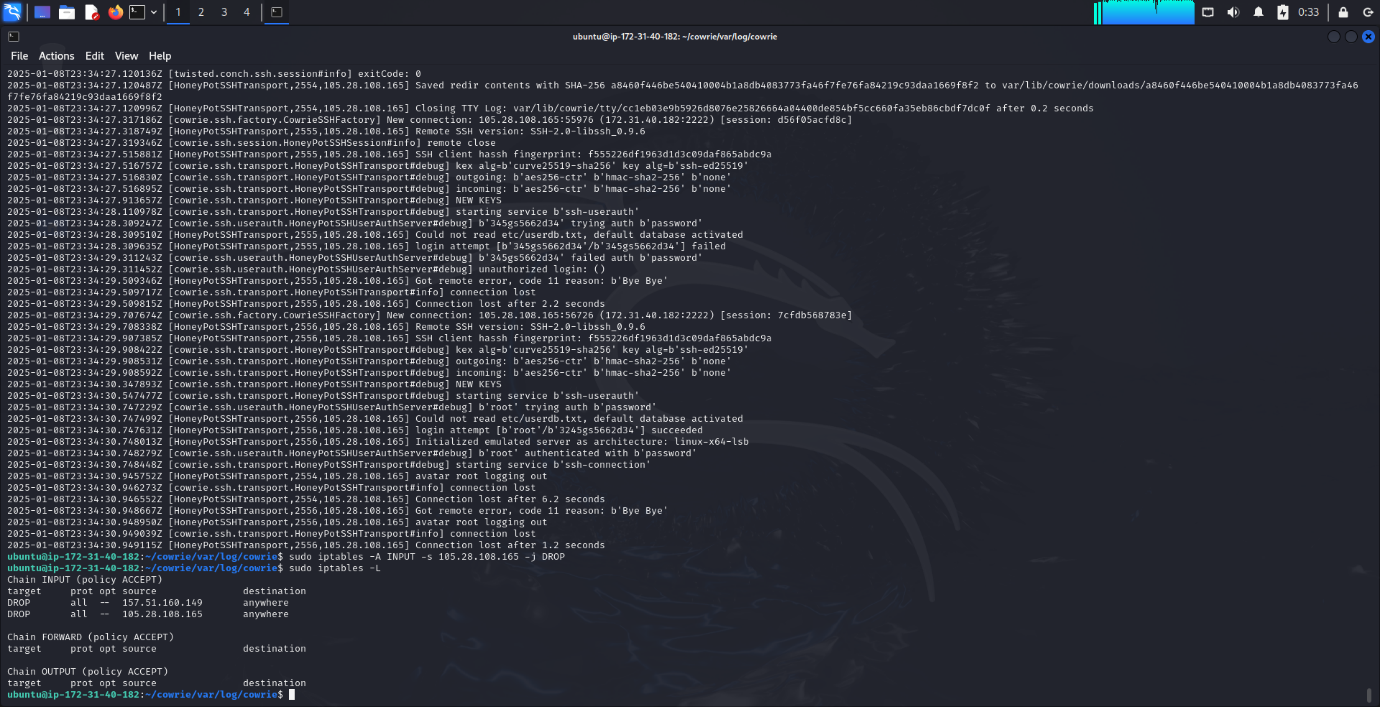
* Honeypot logs located at /home/cowrie/cowrie/var/log/cowrie.log.
* AWS Security Groups for blocking malicious IPs.

**Steps**:

1. Monitored logs in real time:
2. tail -f /home/cowrie/cowrie/var/log/cowrie.log
3. Identified suspicious IPs from the logs.
4. Blocked malicious IPs using AWS CLI:
5. aws ec2 revoke-security-group-ingress --group-id <group-id> --protocol tcp --port 2222 --cidr <malicious-ip>/32
6. Documented attack patterns for analysis.

A screenshot of a computer

AI-generated content may be incorrect.



**Results and Key Insights**

**Honeypot Effectiveness**

* Captured all attack attempts, including commands, IPs, and timestamps.
* Successfully mimicked a vulnerable SSH service to attract attackers.

**Defensive Success**

* Real-time monitoring enabled immediate detection and mitigation of threats.
* Blocking malicious IPs reduced repeated attempts.

**Lessons Learned**

1. Logging provides critical insights into attacker behavior.
2. Honeypots are invaluable for testing real-world defense strategies.
3. Collaboration between attackers and defenders enhances understanding of both offensive and defensive tactics.

**Conclusion**

This project demonstrated the power of honeypots in cybersecurity. Deploying Cowrie on AWS allowed me to study attack patterns and test defense mechanisms in a controlled environment. The experience reinforced the importance of proactive monitoring and logging.